

Amendments to the Claims:

This listing of claims will replace, without prejudice, all prior versions, and listings, of claim in this application.

Listing of Claims:

1. (Presently Amended) A method comprising:
testing a priority inheritance variable associated with a task holding at least one resource; and
while the task still holds the at least one resource, lowering a current priority of the task when testing the priority inheritance variable indicates that the task holds no resources that are involved in a priority inheritance.
2. (Original) The method of claim 1, wherein
the priority inheritance variable is configured to have a value indicative of the number of resources held by the task that higher priority tasks are waiting to receive.
3. (Presently amended) The method of claim 1, further comprising:
raising the current priority of the task when a higher priority task blocks on a second resource held by the task.
4. (Presently amended) The method of claim 3, wherein,
when raising the current priority of the task when the higher priority task blocks on the second resource held by the task, the current priority of the task is raised to a current priority of the higher priority task that blocked on the second resource held by the task.
5. (Presently amended) The method of claim 1, further comprising:
adjusting the priority inheritance variable when a higher priority task blocks on a second resource held by the task.
6. (Presently amended) The method of claim 5, wherein,

when adjusting the priority inheritance variable when the higher priority task blocks on the second resource held by the task, the priority inheritance variable is incremented.

7. (Original) The method of claim 5, further comprising:
adjusting the priority inheritance variable when the higher priority task is deleted.
8. (Original) The method of claim 5, further comprising:
adjusting the priority inheritance variable when the higher priority task times out.
9. (Presently amended) The method of claim 1, further comprising:
adjusting the priority inheritance variable when the task releases [the] a second resource.
10. (Presently amended) The method of claim 9, wherein,
when adjusting the priority inheritance variable when the task releases the second resource, the priority inheritance variable is decremented.
11. (Original) The method of claim 1, wherein
the priority inheritance variable is included in a task control block associated with the task.
12. (Original) A method comprising:
raising a current priority of a task to a current priority of a higher priority task when the higher priority task blocks on a resource held by the task;
incrementing a priority inheritance variable when the higher priority task blocks on the resource held by the task, the priority inheritance variable associated with the task and configured to be indicative of the number of resources held by the task that higher priority tasks are waiting to receive;
decrementing the priority inheritance variable when the task releases the resource that the higher priority task has blocked on;
testing the priority inheritance variable; and
lowering the current priority of the task when testing the priority inheritance

variable indicates that the task holds no resources that are involved in a priority inheritance.

13. (Original) A method comprising:

testing a priority inheritance variable associated with a task, the priority inheritance variable configured to have a value indicative of the number of inversion safe mutual exclusion semaphores held by the task that higher priority tasks are waiting to receive; and

setting a current priority of the task to a base priority value when testing the priority inheritance variable indicates that no higher priority tasks are waiting to receive inversion safe mutual exclusion semaphores held by the task.

14. (Original) The method of claim 13, further comprising:

raising the current priority of the task when a higher priority task blocks on an inversion safe mutual exclusion semaphore held by the task.

15. (Original) The method of claim 14, wherein,

when raising the current priority of the task when the higher priority task blocks on the inversion safe mutual exclusion semaphore held by the task, the current priority of the task is raised to a current priority of the higher priority task that blocked on the inversion safe mutual exclusion semaphore held by the task.

16. (Original) The method of claim 13, further comprising:

adjusting the priority inheritance variable when a higher priority task blocks on an inversion safe mutual exclusion semaphore held by the task.

17. (Original) The method of claim 16, wherein,

when adjusting the priority inheritance variable when the higher priority blocks on the inversion safe mutual exclusion semaphore held by the task, the priority inheritance variable is incremented.

18. (Original) The method of claim 13, further comprising:

adjusting the priority inheritance variable when the task releases an inversion safe mutual exclusion semaphore.

19. (Original) The method of claim 18, wherein,

when adjusting the priority inheritance variable when the task releases the inversion safe mutual exclusion semaphore, the priority inheritance variable is decremented.

20. (Original) The method of claim 13, wherein

the priority inheritance variable is included in a task control block for the task.

21. (Presently Amended) A method comprising:

raising a current priority of a task when a higher priority task blocks on [an] a first inversion safe mutual exclusion semaphore, the first inversion safe mutual exclusion semaphore being held by the task;

incrementing a counter when the higher priority task blocks on the first inversion safe mutual exclusion semaphore, the counter associated with the task and configured to have a value indicative of the number of inversion safe mutual exclusion semaphores held by the task that higher priority tasks are waiting to receive;

decrementing the counter when the task releases the first inversion safe mutual exclusion semaphore; and

while the task still holds a second inversion safe mutual exclusion semaphore, setting the current priority of the task to a base priority value when the counter is decremented to a value that indicates that the task holds no inversion safe mutual exclusion semaphores involved in priority inheritance.

22. (Original) A system comprising:

a task;

a priority inheritance variable, the priority inheritance variable associated with the task and configured to indicate the number of resources that are held by the task and that at least one higher priority task is blocked on; and

a mutual exclusion control mechanism configured to set a current priority of the task to a base priority value when the priority inheritance variable indicates that no higher

priority tasks are blocked on resources held by the task.

23. (Original) The system according to claim 22, wherein
the mutual exclusion control mechanism is configured to increase the current priority of the task when a higher priority task blocks on a resource held by the task.
24. (Original) A system comprising:
 - a task;
 - a priority inheritance variable associated with the task, the variable configured to indicate the number of inversion safe mutual exclusion semaphores that are held by the task and that at least one higher priority task is blocked on; and
 - a mutual exclusion control mechanism configured to set a current priority of the task to a base priority value when the priority inheritance variable indicates that no higher priority tasks are blocked on inversion safe mutual exclusion semaphores held by the task.
25. (Original) The system according to claim 24, wherein
the mutual exclusion control mechanism is configured to increase the current priority of the task when a higher priority task blocks on an inversion safe mutual exclusion semaphore held by the task.
26. (Original) A system comprising:
 - a semaphore; and
 - a variable, the variable associated with the semaphore and configured to indicate whether a pending request for the semaphore has resulted in a priority inheritance.
27. (Original) The system according to claim 26, further comprising:
 - a semaphore control data structure, the semaphore control data structure associated with the semaphore and including the variable.
28. (Original) The system according to claim 26, further comprising:
 - a task;
 - a priority inheritance variable associated with the task; and

a mutual exclusion control mechanism configured to adjust the priority inheritance variable when the task releases the semaphore only if the variable indicates that a pending request for the semaphore has resulted in a priority inheritance.

29. (Original) The system according to claim 26, further comprising:
- a task holding the semaphore,
 - wherein the variable is configured to indicate a count of the number of pending requests for the semaphore made by tasks with higher priority than the task holding the semaphore.
30. (Original) A method comprising:
- tracking a number of resources held by a task that higher priority tasks are presently blocked on, the tracking using only a predetermined amount of memory;
 - raising a current priority of the task when a higher priority task blocks on a resource held by the task; and
 - setting the current priority of the task to a base priority value whenever no higher priority tasks are waiting to receive any of the resources held by the task and the task still holds at least one resource.
31. (Original) A method comprising:
- tracking a number of inversion safe mutual exclusion semaphores held by a task that higher priority tasks are presently blocked on, the tracking using only a predetermined amount of memory;
 - raising a current priority of the task when a higher priority task blocks on an inversion safe mutual exclusion semaphore held by the task; and
 - setting the current priority of the task to a base priority value whenever no higher priority tasks are waiting to receive any of the inversion safe mutual exclusion semaphores held by the task and the task still holds at least one inversion safe mutual exclusion semaphore.
32. (Presently Amended) An article of manufacture comprising a computer-readable medium having stored thereon instructions adapted to be executed by a processor, the instructions which, when executed, define a series of steps to be used to control priority inheritance, said steps comprising:

testing a priority inheritance variable associated with a task holding at least one resource; and

while the task still holds the at least one resource, lowering a current priority of the task when testing the priority inheritance variable indicates that the task holds no resources that are involved in a priority inheritance.

33. (Original) An article of manufacture comprising a computer-readable medium having stored thereon instructions adapted to be executed by a processor, the instructions which, when executed, define a series of steps to be used to control priority inheritance, said steps comprising:

raising a current priority of a task when a higher priority task blocks on an inversion safe mutual exclusion semaphore, the inversion safe mutual exclusion semaphore being held by the task;

incrementing a counter when the higher priority task blocks on the inversion safe mutual exclusion, the counter associated with the task and configured to have a value indicative of the number of inversion safe mutual exclusion semaphores held by the task that higher priority tasks are waiting to receive;

decrementing the counter when the task releases the inversion safe mutual exclusion semaphore; and

setting the current priority of the task to a base priority value when the counter is decremented to a value that indicates that the task holds no inversion safe mutual exclusion semaphores involved in priority inheritance.

34. (Original) An article of manufacture comprising a computer-readable medium having stored thereon instructions adapted to be executed by a processor, the instructions which, when executed, define a series of steps to be used to control priority inheritance, said steps comprising:

tracking a number of resources held by a task that higher priority tasks are presently blocked on, the tracking using only a predetermined amount of memory;

raising a current priority of the task when a higher priority task blocks on a resource held by the task; and

setting the current priority of the task to a base priority value whenever no higher priority tasks are waiting to receive any of the resources held by the task and the task still holds at least one resource.